Quantum Image Edge Detection

Aayush Jalgaonkar, Pratham Maliya, Mentor: Mr. Manuel Rudolph

May 2025

Abstract

Edge detection is a core part of image processing. It is crucial for object recognition, segmentation, and computer vision tasks. Modern day classical methods like Sobel, Canny, and Laplacian operators have been widely used, but they face limitations in scalability, noise sensitivity, and computational efficiency for high-resolution data. This study explores a quantum image processing (QIP) approach to edge detection, which uses quantum parallelism and amplitude encoding to process image data in a quantum circuit framework. We implemented the Quantum Hadamard Edge Detection (QHED) algorithm using the Quantum Probability Image Encoding (QPIE) model and simulated it on IBM Qiskit. Results show that, while current quantum hardware does not yield the best resolution, the quantum method achieved edge detection results comparable to classical methods, with theoretical advantages in speedup and resource compression for large-scale images. Using the findings, this paper critically analyzes the theoretical potential of QIP for scalable image analysis and its practical disadvantages compared to the classical approach.

1 Introduction

Edge Detection in Classical Systems

Edge detection is used to identify boundaries within images. It plays a critical role in applications like facial recognition, medical imaging, autonomous driving, and industrial inspection. Traditional edge detectors—like the Sobel, Prewitt, Canny, and Laplacian operators—work by highlighting regions in an image where intensity changes sharply. However, classical edge detection methods suffer from limitations in speed, scalability, and noise sensitivity, especially in high-resolution or real-time scenarios. These limitations have initiated growing interest in quantum computing's potential to enhance this task.

Edge Detection in Potential Quantum Systems

For the quantum approach of edge detection, we leverage Quantum Image Processing. Quantum Image Processing (QIP) is an emerging field at the intersection of quantum computing and image analysis, offering promising new

paradigms for handling visual data. QIP research and developments demonstrate the feasibility of achieving exponential speedup and data compression, using quantum methods.
Quantum Edge Detectors refer to algorithms or circuits running on quantum hardware that perform edge detection on digital images by leveraging the principles of quantum mechanics—such as superposition, entanglement, and quantum parallelism. The core idea is to exploit the exponential parallelism offered by quantum computation to perform image analysis faster and more efficiently than classical systems.
However, Quantum Image Edge detection (QIED) has some crucial fundamental and practical limitations. Due to hardware constraints, encoding overhead, and algorithmic immaturity, QIED faces a fundamental limitation. The lack of a supportive ecosystem and high operating costs also prevent QIED from becoming a reality. QIP requires significant development and time to be invested in it before its systems could reach theoretical maximum capabilities in multiple domains and make QIED a practical method.

Motivation

Often, we tend to assume quantum computing to be the solution to all the problems classical methods display and we wish to foster more progressive and advanced technologies. But given the current developments, there are many domains where classical computing offers pragmatic advantages over quantum computing. From an operational standpoint, it thus becomes important to use the suitable method for suitable tasks and not employ methods of very high potential to tasks which do not require that much potential.

This paper critically and systematically compares classical models with current quantum models, both in their current forms and in light of their projected future capabilities. Through this comparison, we provide the reader with a clear understanding of the strengths and limitations of each model. This study lays out the specifications, scope, and some important facts about both methods. The study outlines key specifications, experimental results from simulations using IBM's Qiskit platform, and insights into how Quantum Hadamard Edge Detection (QHED), implemented via the Quantum Probability Image Encoding (QPIE) model, performs relative to traditional methods.
Finally, the study presents a perspective on the future of QIED and suggests some steps to be taken to maximize the rate of our progress into Quantum Computing.

2 Key Concepts in Quantum Image Edge Detection

The Quantum Computing methods to processing images are objectively different from the classical methods. The fundamentals of Quantum Image Processing

(QIP) are discussed here before moving into the experimental sections.

Quantum Gates and Circuits are applied on the features which allow us to leverage the advantage of Quantum bits (called qubits). Image data is encoded into the qubits, and specific gates (e.g., Hadamard, CNOT, phase gates) are applied to extract edge features. After quantum operations, the system is measured, and classical post-processing (often light-weight) is applied to interpret the results. This paper primarily deals with experiments using the Quantum Probability Image Encoding (QPIE) and the Quantum Hadamard Edge Detection (QHED) methods.

3 Quantum Probability Image Encoding

Quantum Probability Image Encoding (QPIE) is a method of encoding classical image data into quantum states using probability amplitudes. This encoding uses the principle of amplitude encoding, where grayscale pixel intensities are mapped to the amplitudes of a quantum state vector. The QPIE model allows efficient image compression and parallelism in quantum computation by representing 2^n pixel values using only n qubits [4].

Unlike other models such as FRQI [3] or NEQR [6], which store image information in quantum phases or basis states, the core principle of QPIE is to encode the pixel intensity values of an image into the probability amplitudes of a pure quantum state, while the pixel positions are encoded into the computational basis states of the Hilbert space [4].

For a classical 2D image I of size $N_1 \times N_2$ pixels, the total number of pixels is $N = N_1 \times N_2$. In QPIE, this N-pixel image can be represented using n qubits, where n is given by:

$$n = \lceil \log_2 N \rceil \tag{1}$$

This represents an exponential reduction in the number of fundamental units required for storage compared to classical methods, where $N \times d$ bits would be needed for a d-bit grayscale image.

To convert a classical image into its QPIE quantum state, the 2D pixel data (I_{yx}) is first vectorized into a 1D array of pixel intensities $(I_0, I_1, \ldots, I_{N-1})$. Each pixel intensity I_k (corresponding to pixel at position (y, x) for the k-th element in the vectorized array) must then be normalized to form a valid probability amplitude c_k. This normalization ensures that the sum of the squares of the amplitudes equals 1, a requirement for any quantum state. The normalization is performed as:

$$c_k = \frac{I_k}{\sqrt{\sum_{j=0}^{N-1} I_j^2}} \tag{2}$$

Typically, pixel values must be appropriately scaled before this normalization to ensure the resulting quantum state is properly normalized [4].

Once normalized, the image is represented as a quantum state $|\text{Img}\rangle$ in an n-qubit register:

$$|\text{Img}\rangle = \sum_{k=0}^{N-1} c_k |k\rangle \tag{3}$$

Here, $|k\rangle$ is the computational basis state corresponding to the binary representation of the index k, which encodes the position of the k-th pixel in the vectorized image. The amplitude c_k associated with $|k\rangle$ encodes the normalized intensity of that pixel.

The preparation of such a quantum state can be achieved efficiently. If the image data is stored in a quantum random access memory (qRAM), mapping this data to the quantum state takes $\mathcal{O}(n)$ steps [2, 4]. Furthermore, if the coefficients c_k and the normalization factor $\sum_k |c_k|^2$ can be efficiently computed by a classical algorithm, the construction of the n-qubit image state $|\text{Img}\rangle$ can be performed in $\mathcal{O}(\text{poly}(n))$ steps.

Compared to other QImR models like FRQI and NEQR, QPIE requires fewer qubit resources [4]. For an image with 2^{2m} pixels, FRQI requires $1 + 2m$ qubits and NEQR requires $d + 2m$ qubits (where d is the bit-depth of pixel values), whereas QPIE needs only $2m$ qubits as the pixel value is encoded in the amplitude itself. This efficiency in qubit usage makes QPIE an attractive model for developing QIP algorithms, particularly for resource-constrained near-term quantum devices.

State Preparation in Qiskit

We implement QPIE in Qiskit by using the built-in `initialize` method, which, for an N-dimensional amplitude vector c_k , constructs a depth-optimized set of gates that prepare Equation (3) on $n = \log_2 N$ qubits. In practice, one flattens the normalized $M \times L$ image to a length-N array, then calls:

```
from qiskit import QuantumCircuit
qc = QuantumCircuit(n)
qc.initialize(c_vector, qc.qubits)
```

This requires $O(N)$ classical preprocessing to compute the normalization, and Qiskit's state-preparation routine compiles to $O(n^2\, 2^n)$ elementary gates in the worst case, matching the theoretical bound for arbitrary amplitude encoding.

4 Quantum Hadamard Edge Detection

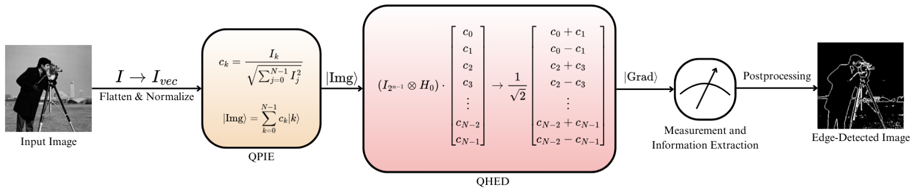
Edge detection is a fundamental task in image processing, aiming to identify significant discontinuities in pixel intensities, which typically correspond to object boundaries or changes in surface texture. Classical edge detection algorithms, such as Sobel, Prewitt, or Canny operators, generally rely on computing image gradients by convolving the image with small filter masks. For an image with

$N = 2^n$ pixels, these methods typically require $\mathcal{O}(N)$ or $\mathcal{O}(2^n)$ operations, as each pixel or its neighborhood needs to be processed.

The Quantum Hadamard Edge Detection (QHED) algorithm, built upon the QPIE representation, offers a quantum approach to this task. QHED uses the properties of the Hadamard gate to compute differences between adjacent pixel intensities in parallel, thereby detecting edges.

The basic action of a Hadamard gate H on a single qubit is:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle),\ H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$



$I \to I_{vec}$
Flatten & Normalize
$c_k = \frac{I_k}{\sqrt{\sum_{j=0}^{N-1} I_j^2}}$
$|\mathrm{Img}\rangle = \sum_{k=0}^{N-1} c_k|k\rangle$
QPIE
$|\mathrm{Img}\rangle$
$(I_{2^{n-1}} \otimes H_0) \cdot$
QHED
$|\mathrm{Grad}\rangle$
Postprocessing
Input Image
Measurement and Information Extraction
Edge-Detected Image

Figure 1: Simplified Quantum Hadamard Edge Detection (QHED) Pipeline. The input image is flattened and normalized for QPIE. The QHED core applies $I_{2^{n-1}} \otimes H_0$ to the QPIE state $|\mathrm{Img}\rangle$, producing a state $|\mathrm{Grad}\rangle$ where amplitudes encode pixel differences. Measurement and classical postprocessing yield the final edge-detected image.

Consider an image encoded using QPIE as $|\mathrm{Img}\rangle = \sum_{k=0}^{N-1} c_k|k\rangle$, where $|k\rangle$ represents the n-qubit computational basis state for pixel position k. For any pair of neighboring pixels whose positions differ only in the least significant bit (LSB), e.g., $|b_{n-1}\dots b_1 0\rangle$ and $|b_{n-1}\dots b_1 1\rangle$, their corresponding amplitudes are $c_{\dots 0}$ and $c_{\dots 1}$. Applying a Hadamard gate to only the LSB (qubit q_0) of the n-qubit register, represented by the unitary $U_H = I_{2^{n-1}} \otimes H_{q_0}$, transforms the input state vector [4] (Equations 8 and 9).

$$(I_{2^{n-1}} \otimes H_0) : \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{N-2} \\ c_{N-1} \end{bmatrix} \to \frac{1}{\sqrt{2}} \begin{bmatrix} c_0 + c_1 \\ c_0 - c_1 \\ c_2 + c_3 \\ c_2 - c_3 \\ \vdots \\ c_{N-2} + c_{N-1} \\ c_{N-2} - c_{N-1} \end{bmatrix}$$

The term $(c_i - c_{i+1})$ represents the gradient between these adjacent pixels. If the LSB q_0 is measured to be in state $|1\rangle$, the post-measurement state will have amplitudes proportional to these differences. This procedure, as described, yields gradients for "even-odd" pixel pairs (e.g., 0-1, 2-3, etc.) along one scanning direction (e.g., horizontal if the image is vectorized column-major).

The overall pipeline for this simpler QHED variant, from input image to edge-detected output, is illustrated in Figure 1.

4.1 QHED with an Ancillary Qubit

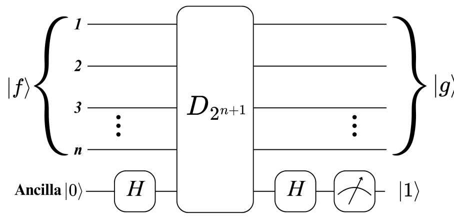


Figure 2: Quantum circuit for the QHED algorithm with an ancillary qubit

To obtain all edge information (i.e., gradients between all adjacent pixel pairs, including "odd-even" pairs like 1-2, 3-4, etc.) in a more comprehensive manner, Yao et al. [4] (Appendix E) and the Qiskit textbook [1] outline a variation of QHED using an ancillary qubit.

1. Initialization: Start with an $(n+1)$-qubit state, where n qubits encode the image $|\text{Img}\rangle$ using QPIE, and one ancillary qubit is initialized to $|0\rangle_A$. The initial state is $|\text{Img}\rangle \otimes |0\rangle_A$.

2. First Hadamard Gate: Apply a Hadamard gate to the ancillary qubit (H_A). Assuming the ancilla is the LSB of the combined $(n+1)$-qubit system, the state vector amplitudes become $\frac{1}{\sqrt{2}}(c_0, c_0, c_1, c_1, \ldots, c_{N-1}, c_{N-1})^T$. This is referred to as a "redundant image state" [4].

3. Amplitude Permutation: Apply a specific $(n+1)$-qubit unitary operation $D_{2^{n+1}}$ to this state.

$$D_{2^{n+1}} = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

This operator is effectively a controlled cyclic shift or a "decrement gate" acting on the computational basis states of the $(n+1)$ qubits. It transforms the amplitude vector to $\frac{1}{\sqrt{2}}(c_0, c_1, c_1, c_2, c_2, c_3, \ldots, c_{N-2}, c_{N-1}, c_{N-1}, c_0)^T$. This permutation can be implemented efficiently in $\mathcal{O}(\text{poly}(n+1))$ time [4].

4. Second Hadamard Gate: Apply a Hadamard gate to the ancillary qubit (H_A) again, we obtain the gradients for both even- and odd-pixel-pairs at the same time.

$$(I_{2^n} \otimes H) \cdot \begin{bmatrix} c_0 \\ c_1 \\ c_1 \\ c_2 \\ c_2 \\ c_3 \\ \vdots \\ c_{N-2} \\ c_{N-1} \\ c_{N-1} \\ c_0 \end{bmatrix} \rightarrow \begin{bmatrix} c_0 + c_1 \\ c_0 - c_1 \\ c_1 + c_2 \\ c_1 - c_2 \\ c_2 + c_3 \\ c_2 - c_3 \\ \vdots \\ c_{N-2} + c_{N-1} \\ c_{N-2} - c_{N-1} \\ c_{N-1} + c_0 \\ c_{N-1} - c_0 \end{bmatrix}$$

5. Measurement: Measure the ancillary qubit. If the outcome is $|1\rangle_A$, the remaining n qubits will be in a state where the amplitudes encode all the desired pixel intensity differences (gradients).

This process provides the edge information along one dimension (e.g., horizontal edges). To detect edges in a 2D image comprehensively, the QHED algorithm must be performed twice. Once on the original image data (e.g., vectorized column-by-column) to detect horizontal edges, and once on the transposed image data I^T to detect vertical edges. The final edge-detected image is then constructed by classically combining the magnitudes of these horizontal and vertical gradient components [4].

The core processing step of QHED (the Hadamard gate applications and the permutation $D_{2^{n+1}}$) can be performed in $\mathcal{O}(\text{poly}(n))$ time, where $n = \log_2 N$. This suggests a potential exponential speedup over classical algorithms for the gradient computation part, especially for large images, as the quantum operations act on all pixel information encoded in superposition simultaneously. However, state preparation and full state readout (measurement) also contribute to the overall complexity and are subjects of ongoing research for practical QIP implementations.

Circuit Implementation

In our Qiskit implementation, $D_{2^{n+1}}$ is realized as a one-position cyclic permutation over the full 2^{n+1}-dimensional Hilbert space:

```
D2n_1 = np.roll(np.identity(2(n+1)), 1, axis=1)
qc = QuantumCircuit(n+1)
qc.initialize(c_vector, range(1, n+1))
qc.h(0)
qc.unitary(D2n_1, range(n+1))
qc.h(0)
```

We use the `statevector_simulator` to extract the amplitudes directly. The amplitudes of basis states with ancilla $|1\rangle$ represent the magnitude of local intensity differences. Thresholding these amplitudes gives a binary edge map for each $w \times w$ image patch.

5 Experiment and Results

To evaluate the performance of the Quantum Hadamard Edge Detection (QHED) algorithm using the Quantum Probability Image Encoding (QPIE) model, we conducted a series of simulations using IBM's Qiskit and the Qiskit Aer statevector simulator. The experiments focused on applying QHED to grayscale images and assessing the quality of the resulting edge maps in comparison to established classical edge detection methods.

5.1 QHED Implementation and Patch-Based Processing

The QHED algorithm, as detailed in Section 3.1, was implemented following the ancillary qubit approach. For practical simulation on current classical hardware, which has limitations in handling the exponentially growing statevector size for large quantum systems, we adopted a patch-based processing strategy for larger images. An input image of size 256×256 pixels was divided into smaller, non-overlapping square patches. The QHED algorithm was then applied independently to each patch.

Both the patch and its transpose were separately amplitude-encoded using the QPIE method to prepare the quantum states for horizontal and vertical edge detection, respectively.

Two quantum circuits were constructed for each patch. The first circuit, for horizontal edge detection, initialized n data qubits (where 2^n is the number of pixels in the patch) with the QPIE state of the patch and utilized one ancillary qubit. It then applied a Hadamard gate to the ancilla, the $D_{2^{n+1}}$ amplitude permutation unitary, and another Hadamard to the ancilla. The second circuit performed the same operations but was initialized with the QPIE state of the transposed patch for vertical edge detection.

The statevector of each circuit was obtained using the Qiskit Aer statevector simulator. From the resulting statevector, amplitudes corresponding to the ancillary qubit being in state $|1\rangle$ were extracted. These amplitudes, representing pixel intensity gradients, were then thresholded: if the absolute value of the real part of an amplitude exceeded a predefined threshold, the corresponding output pixel was set to 1 (edge); otherwise, it was set to 0 (no edge). The thresholds were empirically chosen for each image to produce visually reasonable edge maps.

The binary edge maps obtained from the horizontal and vertical scans of each patch were combined using a logical OR operation. Finally, these processed patches were reassembled to form the complete edge-detected image.

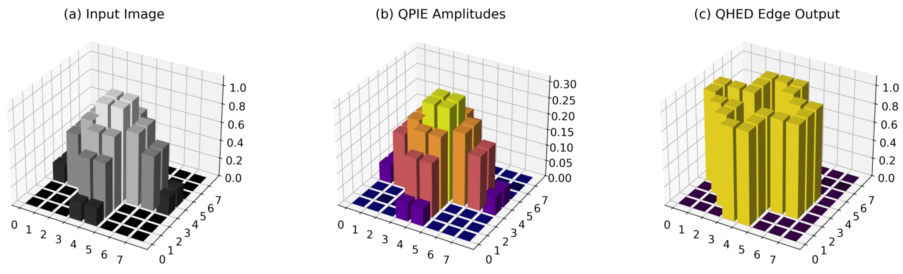


Figure 3: Illustration of QHED on an 8×8 image: (a) Normalized input image, (b) Corresponding QPIE probability amplitudes, (c) QHED edge output after processing and thresholding.

An illustrative example of this process on a smaller 8×8 synthetic image is shown in Figure 3, depicting the input image, its QPIE amplitude representation, and the final QHED output.

5.2 Qualitative Results on Standard Test Images

We applied the patch-based QHED algorithm to three standard 256×256 grayscale test images: "mountain.jpg", "lena.jpg", and "camera.jpg". For "mountain.jpg", a patch size of 32×32 pixels and an amplitude threshold of 1×10^{-3} were used. For "lena.jpg" and "camera.jpg", a patch size of 16×16 pixels and an amplitude threshold of 1×10^{-2} were employed to balance computational feasibility with result quality. The qualitative results, comparing the original image with the QHED output, as well as outputs from classical Canny and Sobel edge detectors, are presented in Figures 4, 5, and 6.

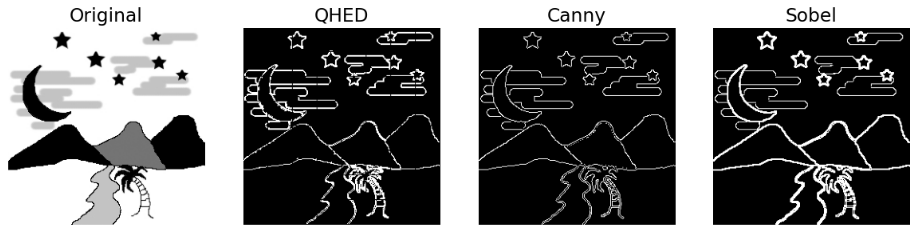


Figure 4: Edge detection comparison for "mountain.jpg" (256×256). From left to right: Original image, QHED output (patch size 32×32, threshold 1×10^{-3}), Canny edge detector, Sobel edge detector.

Visually, the QHED algorithm successfully identifies the prominent edges in all three images. The detected edges are generally consistent with those found by the classical Sobel and Canny operators. However, due to the patch-based approach and the nature of amplitude encoding and thresholding, some blockiness or grid artifacts are noticeable in the QHED outputs, particularly

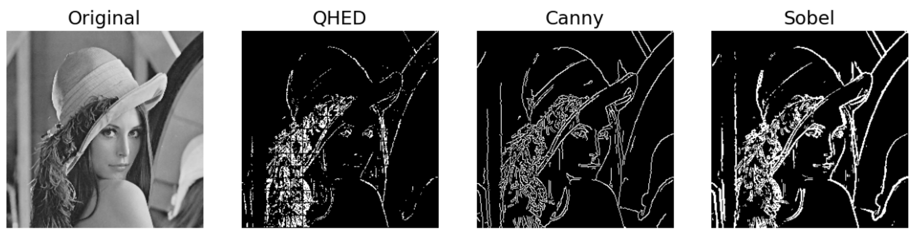


Figure 5: Edge detection comparison for "lena.jpg" (256×256). From left to right: Original image, QHED output (patch size 16×16, threshold 1×10^{-2}), Canny edge detector, Sobel edge detector.

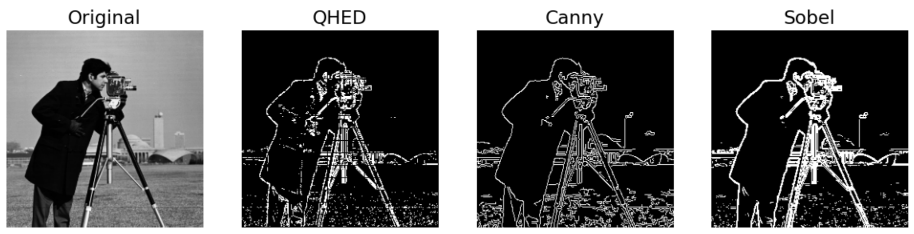


Figure 6: Edge detection comparison for "camera.jpg" (256×256). From left to right: Original image, QHED output (patch size 16×16, threshold 1×10^{-2}), Canny edge detector, Sobel edge detector.

where patch boundaries coincide with subtle intensity gradients. The choice of threshold significantly impacts the sensitivity of the QHED method; a lower threshold tends to detect more, potentially noisy, edges, while a higher threshold retains only stronger edges.

5.3 Quantitative Evaluation: Peak Signal-to-Noise Ratio (PSNR)

To quantitatively assess the quality of the edge maps produced by QHED, we calculated the Peak Signal-to-Noise Ratio (PSNR) using the binary edge maps from the classical Sobel and Canny algorithms as reference (ground truth). The PSNR was evaluated for varying patch sizes (2×2, 4×4, 8×8, 16×16, and 32×32), which correspond to total qubit counts (data qubits + 1 ancilla qubit) of 3, 5, 7, 9, and 11, respectively. The results are plotted in Figure 7.

As shown in Figure 7, the PSNR generally increases with the number of qubits, which corresponds to larger patch sizes. This trend suggests that processing larger regions of the image quantumly at once (i.e., using larger patches) allows for a more coherent and accurate representation of edges, leading to results that are quantitatively more similar to the classical outputs. The "mountain.jpg" image consistently shows higher PSNR values, possibly due to its

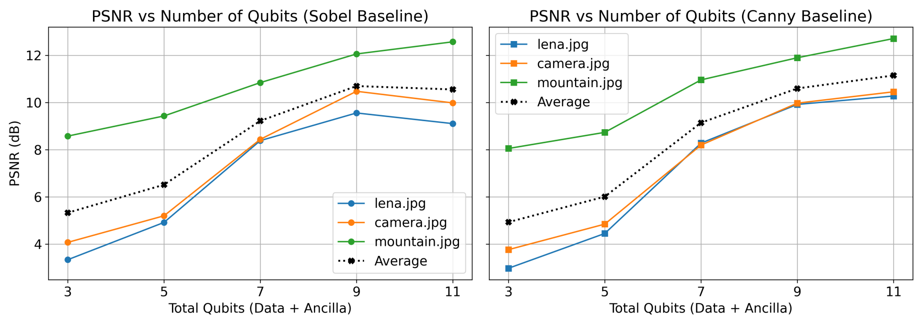


Figure 7: Peak Signal-to-Noise Ratio (PSNR) of QHED outputs compared to classical baselines, as a function of the total number of qubits used per patch. Left: PSNR with Sobel edge map as reference. Right: PSNR with Canny edge map as reference. Individual lines represent different test images, and the black dotted line shows the average PSNR.

distinct and less complex edge structures compared to "lena.jpg" and "camera.jpg". While the PSNR values do not reach levels indicating perfect replication of classical methods, the increasing trend with patch size (and thus qubit count) highlights the potential for improved performance as more quantum resources become available for encoding larger image segments. The PSNR values are generally higher when compared against the Sobel detector than the Canny detector, which might be attributed to Sobel being a simpler gradient-based operator, conceptually closer to the fundamental operation of QHED. Canny, on the other hand, involves more sophisticated post-processing steps like non-maximum suppression and hysteresis thresholding, which are not directly mirrored in the current QHED implementation.

6 Complexity Analysis

The theoretical advantage of quantum algorithms often lies in their potential for exponential speedups over classical counterparts. For Quantum Image Processing (QIP), and specifically for the Quantum Hadamard Edge Detection (QHED) algorithm, complexity analysis involves considering qubit resources, gate counts, circuit depth, and the scaling of these parameters with image size.

6.1 Qubit Resources

Classical edge detection algorithms operate on images with N pixels and typically have a time complexity in the order of $\mathcal{O}(N)$ (or $\mathcal{O}(2^n)$ where $N = 2^n$) for worst-case scenarios, with some improved techniques achieving $\mathcal{O}(N \log N)$ (or $\mathcal{O}(mn \log(mn))$ for an $m \times n$ image.

In the quantum domain, various QImRs have been proposed. The QSobel algorithm, for instance, utilizes the Flexible Representation of Quantum Images

(FRQI) [3, 5]. FRQI encodes an $N \times N$ image (where $N = 2^k$ for k qubits per dimension) using $1 + 2k$ qubits if pixel values are encoded in an angle, or more if values are encoded in basis states. While QSobel offers a theoretical speedup with a complexity of $\mathcal{O}(k^2)$ (where $k = \log_2 N$), FRQI's state preparation can be complex (circuit depth of $\mathcal{O}(k) + \mathcal{O}(\log^2 k)$ in the worst case) and requires more qubits than QPIE, which is a limited resource on current hardware. Moreover, QSobel relies on certain subroutines like a COPY operation and a quantum black box for gradient calculation, for which efficient general implementations are challenging [4].

The QHED algorithm, as employed in this study, uses the Quantum Probability Image Encoding (QPIE) model. As detailed in Section 2, QPIE represents an N-pixel image (or image patch) using $n = \lceil \log_2 N \rceil$ data qubits. The QHED algorithm with the ancillary qubit approach requires one additional qubit, totaling $n+1$ qubits. This logarithmic scaling offers an exponential decrease in qubit resources compared to FRQI or NEQR for storing the same image information [4].

6.2 State Preparation and Core QHED Operations

The Qiskit textbook notes that the time complexity for image encoding using QPIE is $\mathcal{O}(n^2)$ [1] , which is slightly higher than the state preparation part of FRQI. Constructing the n-qubit QPIE state can be done in $\mathcal{O}(\text{poly}(n))$ steps if the pixel amplitudes and normalization factor are efficiently classically computable [4].

The QHED algorithm itself involves two Hadamard gates on the ancillary qubit and one application of the $(n + 1)$-qubit amplitude permutation unitary $D_{2^{n+1}}$. The Hadamard gates are $\mathcal{O}(1)$. The $D_{2^{n+1}}$ unitary (decrement gate) has a circuit depth of $\mathcal{O}(\text{poly}(n+1))$. Crucially, because QHED smartly utilizes the interference property of the Hadamard gate for gradient computation, the core edge detection procedure (excluding state preparation and the full permutation) can be considered highly efficient. The Qiskit textbook and Yao et al. suggest a time complexity of $\mathcal{O}(1)$ for this differencing step itself (without including the state preparation and the amplitude permutation unitary) [4, 1]. This is significantly lower than the $\mathcal{O}(n^2)$ complexity of the QSobel algorithm's processing stage.

Theoretically, this gives QHED a superexponential speedup over classical algorithms and a polynomial speedup over QSobel for the edge detection logic itself, provided efficient state preparation and readout.

6.3 Circuit Depth and Gate Counts from Simulation

To analyze the practical complexity of our QHED implementation, we simulated the circuit for various patch sizes ($w \times w$), corresponding to total qubit counts of $n_{total} = \log_2(w^2) + 1$. The circuits, using Qiskit's 'initialize' and 'unitary' gates, were transpiled to '['u3', 'cx']' with optimization level 0.

Figure 8 shows the circuit depth and the number of U3 and CNOT gates versus the total number of qubits.

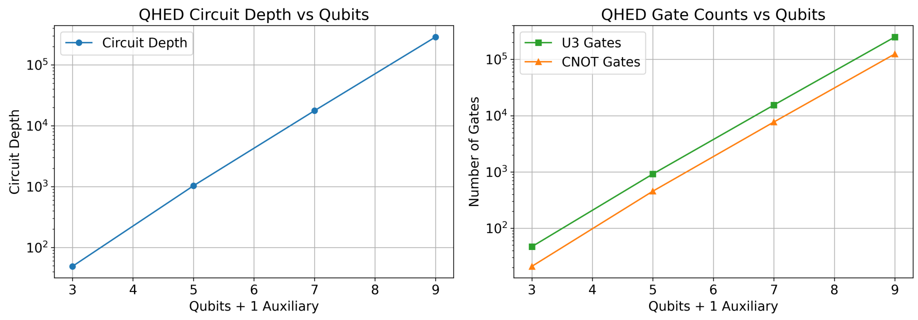


Figure 8: QHED circuit resource scaling with the total number of qubits (data qubits + 1 ancilla). Left: Circuit depth. Right: Number of U3 (single-qubit) and CNOT (two-qubit) gates. Both y-axes are on a logarithmic scale.

As observed in Figure 8, both circuit depth and the number of U3 and CNOT gates increase exponentially with the number of qubits. This exponential growth in the full circuit's gate counts and depth is largely due to the general-purpose 'initialize' function for QPIE state preparation and the 'unitary' gate for the $D_{2^{n+1}}$ permutation when not specifically optimized. While the fundamental edge detection logic via Hadamard transform is simple, the overhead of these general state preparation and unitary operations for arbitrary n dominates the observed complexity in these simulations.

6.4 Measurement Complexity

A critical consideration for practical quantum advantage is the measurement process. To fully reconstruct the classical edge map, which contains N_p pixels in the patch (where $n = \log_2 N_p$ data qubits are used), one would generally need $\mathcal{O}(2^n)$ or $\mathcal{O}(N_p)$ measurements to obtain the probability amplitudes with sufficient precision. This exponential measurement requirement can negate computational speedups if the complete classical output is needed.

However, as noted by Yao et al. [4] and the Qiskit textbook [1], if the objective is not full image reconstruction but rather the identification of specific patterns or global features, a measurement of a single local observable might suffice with a much smaller number of measurements, potentially on the order of $\mathcal{O}(n^2)$. This distinction is important: the limitation is often not inherent to the core quantum algorithm but to the nature of extracting classical information from a quantum state. Thus, a direct comparison of time complexity bounds between quantum and classical edge detection algorithms must account for the specific task and output requirements. For our simulations, the statevector simulator bypasses this by providing direct access to amplitudes.

In summary, QPIE provides exponential compression in qubit resources. The core QHED logic uses quantum parallelism for a $\mathcal{O}(1)$ (or $\mathcal{O}(\text{poly}(n))$ for the permutation part) processing step. However, the overall simulated circuit complexity is dominated by the unoptimized state preparation and general unitary application, leading to an observed exponential scaling of gates with qubit count. Optimized subroutines or a focus on tasks not requiring full state reconstruction are key to realizing the theoretical speedups of QHED.

7 Conclusion

This study investigated the Quantum Hadamard Edge Detection (QHED) algorithm, implemented using the Quantum Probability Image Encoding (QPIE) model, and simulated its performance against classical edge detection methods. Our findings indicate that while QPIE offers significant advantages in qubit resource compression ($\mathcal{O}(\log N + 1)$ qubits for an N-pixel image), and the core QHED logic theoretically promises efficient processing ($\mathcal{O}(1)$ or $\mathcal{O}(\text{poly}(\log N))$ for the differencing step), practical implementation using current quantum simulation frameworks reveals challenges.

The qualitative results from applying QHED to 256×256 grayscale images via a patch-based approach demonstrated that QHED can successfully identify prominent edges, with visual outcomes generally consistent with classical Sobel and Canny operators. However, artifacts such as blockiness due to the patching strategy and sensitivity to amplitude thresholding were observed. Quantitatively, the Peak Signal-to-Noise Ratio (PSNR) of QHED outputs, when compared to Sobel and Canny baselines, showed an encouraging trend of improvement with increasing patch size (and thus qubit count per patch), suggesting better coherence with larger quantum-processed segments.

The complexity analysis highlighted a critical distinction: while the underlying quantum parallelism of QHED is efficient, the simulated full circuit complexity, dominated by general-purpose state preparation (Qiskit's 'initialize') and unitary application, scaled exponentially with the number of qubits per patch. This translates to a polynomial complexity in the number of patch pixels, which, without significant optimization, does not currently outperform classical $\mathcal{O}(N)$ algorithms for the patch itself. Furthermore, the $\mathcal{O}(N)$ measurement complexity for full classical reconstruction of the edge map remains a significant bottleneck for realizing overall quantum speedup if the entire edge map is the desired output.

In its current simulated form, and considering the overheads of general quantum state preparation and unitary operations, QHED does not yet demonstrate a practical performance advantage over mature classical edge detection algorithms for real-world applications. The theoretical potential for speedup in QIP, particularly for edge detection, is contingent upon the development of highly optimized quantum subroutines for image encoding and information extraction, or a shift towards quantum tasks that do not require full classical output reconstruction. While QIP, including QHED, holds great theoretical promise for an era of fault-tolerant and scalable quantum hardware, classical image processing remains superior for current practical applications.

Future Work

Future research in Quantum Image Processing, and specifically Quantum Edge Detection, should focus on several key areas to bridge the gap between theoretical potential and practical application. Algorithmically, efforts are needed to develop more efficient and less resource-intensive quantum image encoding schemes beyond current QPIE, FRQI, and NEQR models, potentially exploring hybrid classical-quantum encoding or adaptive representations. For QHED itself, optimizing the $D_{2^{n+1}}$ amplitude permutation unitary for specific circuit architectures and developing more sophisticated quantum native thresholding or feature extraction techniques post-gradient computation are crucial. Extending these quantum edge detection principles to handle color images, video streams, and 3D volumetric data will expand their applicability. Furthermore, a significant direction involves designing QIP tasks where the quantum advantage is less reliant on full classical state reconstruction, such as quantum-enhanced feature extraction for subsequent classical machine learning or direct quantum image matching and pattern recognition. Studying how well these algorithms handle noise on real NISQ hardware, and building quantum-classical systems that smartly divide tasks between quantum and classical processors, will be key to showing real advantages in image analysis.

8 Acknowledgments

The authors would like to thank Mr. Manuel Rudolph for his valuable insights and guidance throughout this research as a mentor. We also acknowledge the support of the BeyondQuantum team (part of ThinkingBeyond) and Dr. Filip Bar who provided us access to the prerequisite courses, computational resources and a platform to conduct this research.

References

[1] various authors. *Qiskit Textbook*. Github, 2023. URL: `https://github.com/Qiskit/textbook`.

[2] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. “Quantum Random Access Memory”. In: *Physical Review Letters* 100.16 (Apr. 2008). ISSN: 1079-7114. DOI: `10.1103/physrevlett.100.160501`. URL: `http://dx.doi.org/10.1103/PhysRevLett.100.160501`.

[3] Phuc Q. Le, Fangyan Dong, and Kaoru Hirota. "A flexible representation of quantum images for polynomial preparation, image compression, and processing operations". In: *Quantum Information Processing* 10.1 (Feb. 1, 2011), pp. 63–84. ISSN: 1573-1332. DOI: 10.1007/s11128-010-0177-y. URL: https://doi.org/10.1007/s11128-010-0177-y.

[4] Xi-Wei Yao et al. "Quantum Image Processing and Its Application to Edge Detection: Theory and Experiment". In: *Phys. Rev. X* 7 (3 Sept. 2017), p. 031041. DOI: 10.1103/PhysRevX.7.031041. URL: https://link.aps.org/doi/10.1103/PhysRevX.7.031041.

[5] Yi Zhang, Kai Lu, and YingHui Gao. "QSobel: A novel quantum image edge extraction algorithm". In: *Science China Information Sciences* 58 (Dec. 2014). DOI: 10.1007/s11432-014-5158-9.

[6] Yi Zhang et al. "NEQR: a novel enhanced quantum representation of digital images". In: *Quantum Information Processing* 12.8 (Aug. 1, 2013), pp. 2833–2860. ISSN: 1573-1332. DOI: 10.1007/s11128-013-0567-z. URL: https://doi.org/10.1007/s11128-013-0567-z.